K. J. Somaiya College of Engineering

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

Subject: Analysis of Algorithms (AOA) Subject Code: 2UCC402

Term: Even (2020-21) Class/ Sem: S.Y Comp/ IV

(Programming Language – JAVA)

	Algorithms	**Course Outcomes**
Introduction to analysis of algorithm:	-Selection sort -Insertion sort	CO1
Divide and Conquer	-Binary search/ Finding Minimum And Maximum -Merge Sort Analysis /Quick Sort Analysis	CO2
Greedy Method	- Single Source Shortest Path -Knapsack Problem	CO2
Dynamic Programming	- Travelling Salesman Problem -All Pair Shortest Path	CO2
Backtracking	-8 Queens Problem -Sum of Subsets	CO2
String Matching Algorithms	-Longest Common Subsequence Algorithm	CO3

Subject In-charges:
Prof. Ruchika Patil
Prof. Gopal Sonune

K. J. Somaiya College of Engineering

(A Constituent College of Somaiya Vidyavihar University)

Batch: A1 **Roll No.: 1911013**

Experiment No.__1__

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of selection sort/ Insertion sort

Objective: To analyse performance of sorting methods

CO to be achieved:

Sr. No	Objective
CO 1	Analyze the asymptotic running time and space complexity of algorithms.
CO 2	Describe various algorithm design strategies to solve different problems and analyse Complexity.
CO 3	Develop string matching techniques
CO 4	Describe the classes P, NP, and NP-Complete

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://en.wikipedia.org/wiki/Insertion_sort**

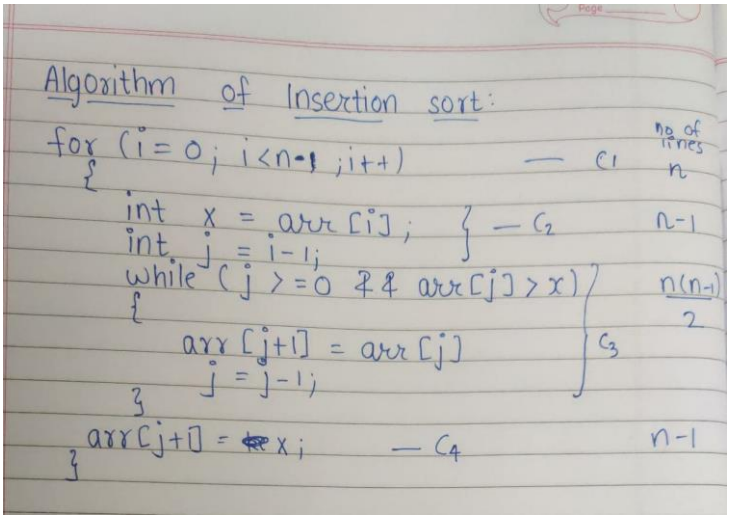
4. **http://www.sorting-algorithms.com/insertion-sort**
5. **http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Insertion_sort.html**
6. **http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/insertionSort.htm**
7. **http://en.wikipedia.org/wiki/Selection_sort**
8. **http://www.sorting-algorithms.com/selection-sort**
9. **http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/selectionSort.htm**
10. **http://courses.cs.vt.edu/~csonline/Algorithms/Lessons/SelectionCardSort/selectioncardsort.html**

Pre Lab/ Prior Concepts:
Data structures, sorting techniques

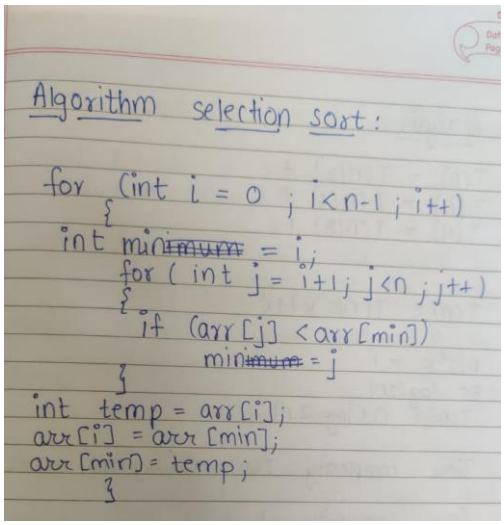
Historical Profile:
There are various methods to sort the given list. As the size of input changes, the performance of these strategies tends to differ from each other. In such case, the priori analysis can helps the engineer to choose the best algorithm.

New Concepts to be learned:
Space complexity, time complexity, size of input, order of growth.

Algorithm InsertionSort



Algorithm SelectionSort



Implementation details:

Selection Sort:

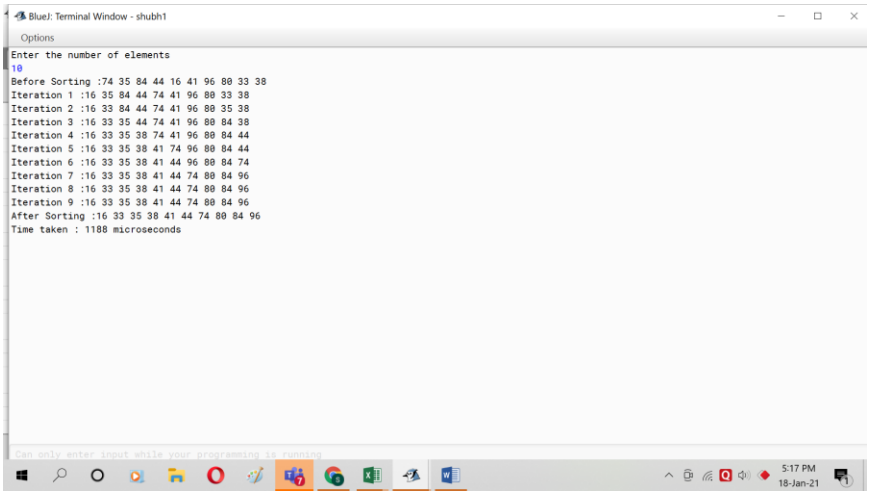
Code:

```
import java.util.*;

import java.lang.Math;

public class Selection_Sort{

   public static void main(String args[])

   {

      Scanner sc = new Scanner(System.in);

      int n,i,z;

      System.out.println("Enter the number of elements");
```

```
n= sc.nextInt();

int arr[]= new int[n];

int rand = 0;

int max = 100;

int min = 1;

long start=0,end=0;

int range = max - min + 1;

for ( i = 0; i < n; i++) {

   rand = (int)(Math.random() * range) + min;

   arr[i]=rand;

}

System.out.print("Before Sorting :");

for ( z=0; z<n; z++)

   System.out.print(arr[z]+" ");

System.out.println("");

start = System.nanoTime();

 for ( i = 0; i < n-1; i++){

   int min_idx = i;

   for (int j = i+1; j < n; j++)

      if (arr[j] < arr[min_idx])

         min_idx = j;

   int temp = arr[min_idx];
```

```
        arr[min_idx] = arr[i];

        arr[i] = temp;

        System.out.print("Iteration "+(i+1)+" :");

        for ( z=0; z<n; z++)

        System.out.print(arr[z]+" ");

        System.out.println("");

      }

      System.out.print("After Sorting :");

      for ( z=0; z<n; z++)

        System.out.print(arr[z]+" ");

      System.out.println();

      end = System.nanoTime();

      long time = (end - start)/1000000;

      System.out.println("Time taken : "+time+" milliseconds");

    }

}
```

Output:



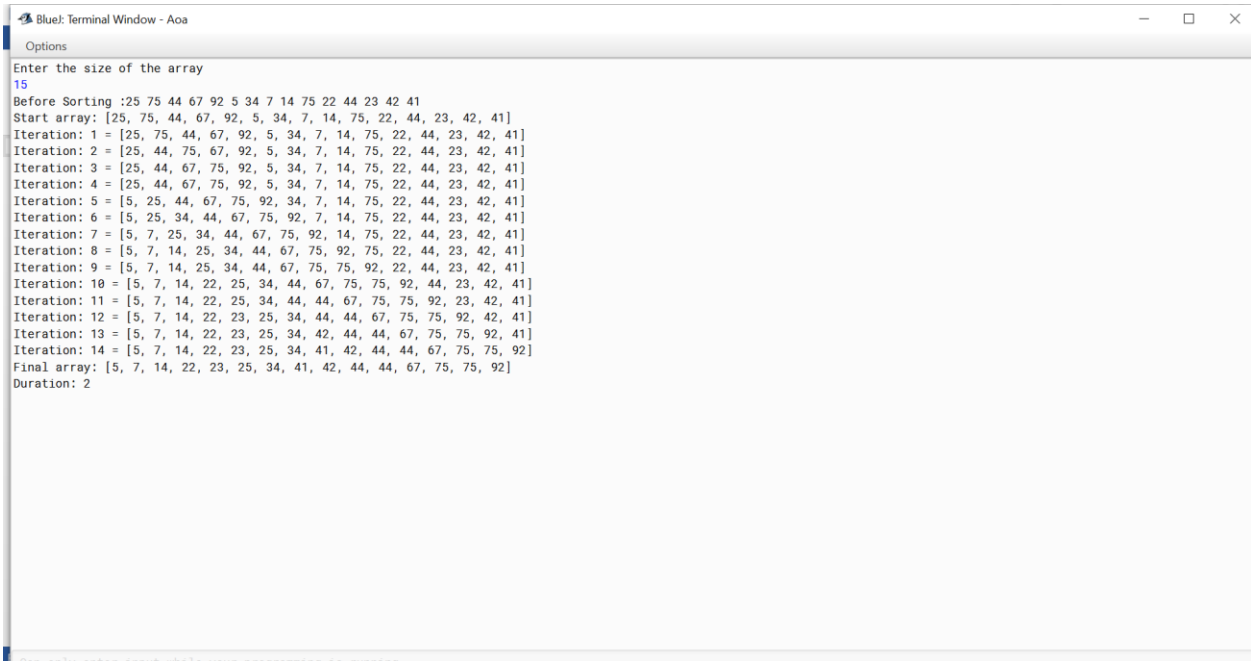
Insertion sort:

Code:

```
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class Insertion_Sort {
    public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
    System.out.println("Enter the size of the array");
    int n =sc.nextInt();
int[] arr = new int[n];
int min=0;
int max=100;
   int range = max - min + 1;
```

```
    for (int i = 0; i < n; i++) {
      int rand = (int)(Math.random() * range) + min;
       arr[i]=rand;
    }
    System.out.print("Before Sorting :");
    for (int z=0; z<n; z++)
       System.out.print(arr[z]+" ");
    System.out.println("");
  System.out.println("Start array: "+ Arrays.toString(arr));
  long startTime = System.currentTimeMillis();
    for(int i=1;i<n;i++){
       int key = arr[i];
       int j = i-1;
       while(j>=0 && arr[j]>key){
          arr[j+1]=arr[j];
          j--;
       }
       arr[j+1] = key;
       System.out.println("Iteration: "+(i)+" = "+Arrays.toString(arr));
    }
  System.out.println("Final array: "+ Arrays.toString(arr));
  long endTime = System.currentTimeMillis();
  long duration = (endTime - startTime);
  System.out.println("Duration: "+duration);
}
}
```

Output:



The space complexity of Insertion sort:

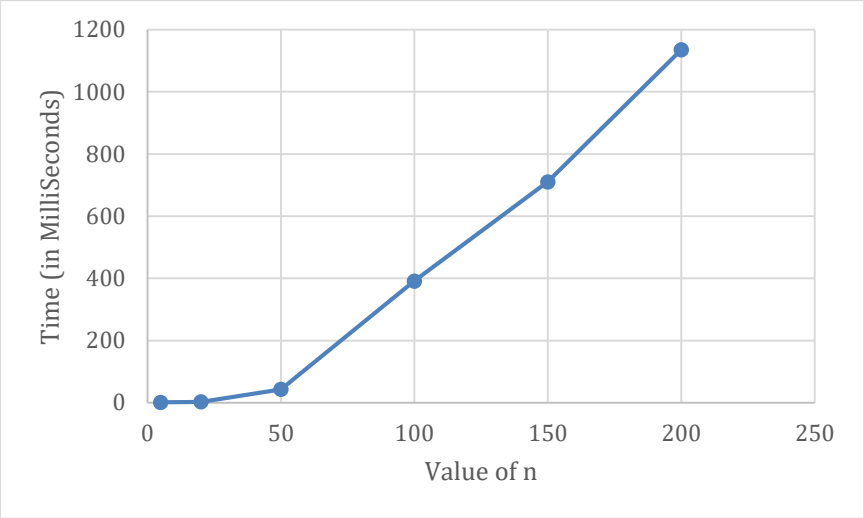
O (1)

The space complexity of Selection sort:

O (1)

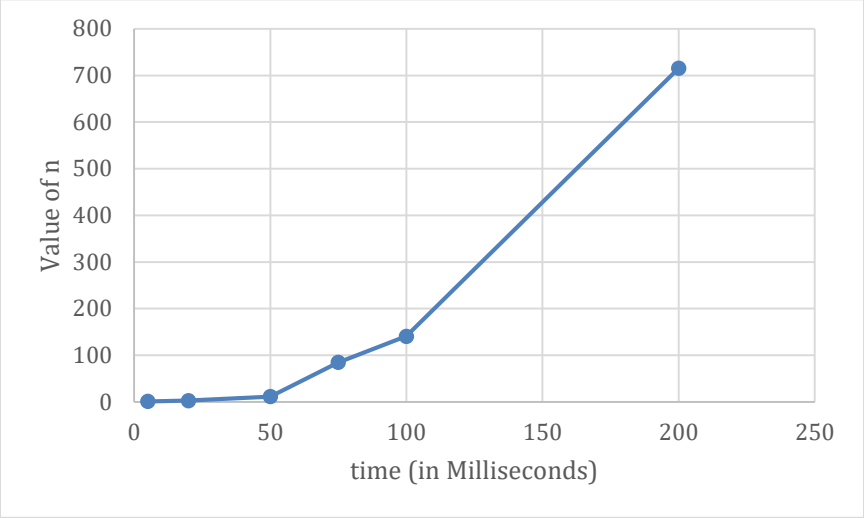
Time complexity for Insertion sort:

O (n^2)

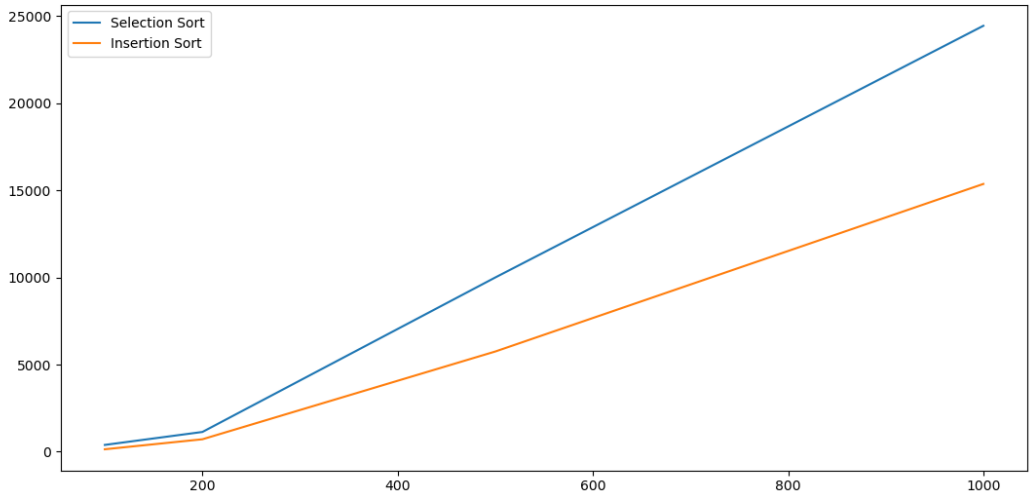


Time complexity for selection sort:

O (n^2)



Graphs for varying input sizes: (Insertion Sort & Selection sort)



CONCLUSION:

Insertion and selection sorts were understood and implemented.
Time and space complexity of both sorting algorithms were understood.